# string manual page - Tcl Built-In Commands

tcl.tk/man/tcl/TclCmd/string.htm

## NAME

string — Manipulate strings

## SYNOPSIS

**string** *option arg* ?*arg ...?*

## DESCRIPTION

Performs one of several string operations, depending on *option*. The legal *option*s (which may be abbreviated) are:

**string cat ?*string1*? ?*string2...*?**

Concatenate the given *string*s just like placing them directly next to each other and return the resulting compound string. If no *string*s are present, the result is an empty string. This primitive is occasionally handier than juxtaposition of strings when mixed quoting is wanted, or when the aim is to return the result of a concatenation without resorting to **return** **-level 0**, and is more efficient than building a list of arguments and using **join** with an empty join string.

**string compare ?-nocase? ?-length *length*? *string1 string2***

Perform a character-by-character comparison of strings *string1* and *string2*. Returns -1, 0, or 1, depending on whether *string1* is lexicographically less than, equal to, or greater than *string2*. If **-length** is specified, then only the first *length* characters are used in the comparison. If **-length** is negative, it is ignored. If **-nocase** is specified, then the strings are compared in a case-insensitive manner.

**string equal ?-nocase? ?-length *length*? *string1 string2***

Perform a character-by-character comparison of strings *string1* and *string2*. Returns 1 if *string1* and *string2* are identical, or 0 when not. If **-length** is specified, then only the first *length* characters are used in the comparison. If **-length** is negative, it is ignored. If **-nocase** is specified, then the strings are compared in a case-insensitive manner.

**string first *needleString haystackString ?startIndex?***

Search *haystackString* for a sequence of characters that exactly match the characters in *needleString*. If found, return the index of the first character in the first such match within *haystackString*. If not found, return -1. If *startIndex* is specified (in any of the forms described in **STRING INDICES**), then the search is constrained to start with the character in *haystackString* specified by the index. For example,

```
string first a 0a23456789abcdef 5
```

will return **10**, but

```
string first a 0123456789abcdef 11
```

will return **-1**.

**string index *string charIndex***
Returns the *charIndex*'th character of the *string* argument. A *charIndex* of 0 corresponds to the first character of the string. *charIndex* may be specified as described in the **STRING INDICES** section.
If *charIndex* is less than 0 or greater than or equal to the length of the string then this command returns an empty string.

**string is *class* ?-strict? ?-failindex *varname*? *string***
Returns 1 if *string* is a valid member of the specified character class, otherwise returns 0. If **-strict** is specified, then an empty string returns 0, otherwise an empty string will return 1 on any class. If **-failindex** is specified, then if the function returns 0, the index in the string where the class was no longer valid will be stored in the variable named *varname*. The *varname* will not be set if **string is** returns 1. The following character classes are recognized (the class name can be abbreviated):

**alnum**
Any Unicode alphabet or digit character.

**alpha**
Any Unicode alphabet character.

**ascii**
Any character with a value less than \u0080 (those that are in the 7-bit ascii range).

**boolean**
Any of the forms allowed to **Tcl_GetBoolean**.

**control**
Any Unicode control character.

**digit**
Any Unicode digit character. Note that this includes characters outside of the [0-9] range.

**double**
Any of the forms allowed to **Tcl_GetDoubleFromObj**.

**entier**
Any of the valid string formats for an integer value of arbitrary size in Tcl, with optional surrounding whitespace. The formats accepted are exactly those accepted by the C routine **Tcl_GetBignumFromObj**.

**false**
Any of the forms allowed to **Tcl_GetBoolean** where the value is false.

**graph**
Any Unicode printing character, except space.

**integer**

Any of the valid string formats for a 32-bit integer value in Tcl, with optional surrounding whitespace. In case of overflow in the value, 0 is returned and the *varname* will contain -1.

**list**
Any proper list structure, with optional surrounding whitespace. In case of improper list structure, 0 is returned and the *varname* will contain the index of the "element" where the list parsing fails, or -1 if this cannot be determined.

**lower**
Any Unicode lower case alphabet character.

**print**
Any Unicode printing character, including space.

**punct**
Any Unicode punctuation character.

**space**
Any Unicode whitespace character, mongolian vowel separator (U+180e), zero width space (U+200b), word joiner (U+2060) or zero width no-break space (U+feff) (=BOM).

**true**
Any of the forms allowed to **Tcl_GetBoolean** where the value is true.

**upper**
Any upper case alphabet character in the Unicode character set.

**wideinteger**
Any of the valid forms for a wide integer in Tcl, with optional surrounding whitespace. In case of overflow in the value, 0 is returned and the *varname* will contain -1.

**wordchar**
Any Unicode word character. That is any alphanumeric character, and any Unicode connector punctuation characters (e.g. underscore).

**xdigit**
Any hexadecimal digit character ([0-9A-Fa-f]).

In the case of **boolean**, **true** and **false**, if the function will return 0, then the *varname* will always be set to 0, due to the varied nature of a valid boolean value.

**string last *needleString haystackString* ?*lastIndex*?**
Search *haystackString* for a sequence of characters that exactly match the characters in *needleString*. If found, return the index of the first character in the last such match within *haystackString*. If there is no match, then return -1. If *lastIndex* is specified (in any of the forms described in **STRING INDICES**), then only the characters in *haystackString* at or before the specified *lastIndex* will be considered by the search. For example,

```
string last a 0a23456789abcdef 15
```

will return **10**, but

```
string last a 0a23456789abcdef 9
```

will return **1**.

**string length *string***
Returns a decimal string giving the number of characters in *string*. Note that this is not necessarily the same as the number of bytes used to store the string. If the value is a byte array value (such as those returned from reading a binary encoded channel), then this will return the actual byte length of the value.

**string map ?-nocase? *mapping string***
Replaces substrings in *string* based on the key-value pairs in *mapping*. *mapping* is a list of *key value key value ...* as in the form returned by **array get**. Each instance of a key in the string will be replaced with its corresponding value. If **-nocase** is specified, then matching is done without regard to case differences. Both *key* and *value* may be multiple characters. Replacement is done in an ordered manner, so the key appearing first in the list will be checked first, and so on. *string* is only iterated over once, so earlier key replacements will have no affect for later key matches. For example,

```
string map {abc 1 ab 2 a 3 1 0} 1abcaababcababababc
```

will return the string **01321221**.

Note that if an earlier *key* is a prefix of a later one, it will completely mask the later one. So if the previous example is reordered like this,

```
string map {1 0 ab 2 a 3 abc 1} 1abcaababcababababc
```

it will return the string **02c322c222c**.

**string match ?-nocase? *pattern string***
See if *pattern* matches *string*; return 1 if it does, 0 if it does not. If **-nocase** is specified, then the pattern attempts to match against the string in a case insensitive manner. For the two strings to match, their contents must be identical except that the following special sequences may appear in *pattern*:
***
Matches any sequence of characters in *string*, including a null string.

**?**
Matches any single character in *string*.

**[*chars*]**
Matches any character in the set given by *chars*. If a sequence of the form *x*-*y* appears in *chars*, then any character between *x* and *y*, inclusive, will match. When used with **-nocase**, the end points of the range are converted to lower case first. Whereas {[A-z]} matches “_” when matching case-sensitively (since “_” falls between the “Z” and “a”), with **-nocase** this is considered like {[A-Za-z]} (and probably what was meant in the first place).

***x***

Matches the single character *x*. This provides a way of avoiding the special interpretation of the characters ***?[]** in *pattern*.

**string range *string first last***
Returns a range of consecutive characters from *string*, starting with the character whose index is *first* and ending with the character whose index is *last*. An index of 0 refers to the first character of the string. *first* and *last* may be specified as for the **index** method. If *first* is less than zero then it is treated as if it were zero, and if *last* is greater than or equal to the length of the string then it is treated as if it were **end**. If *first* is greater than *last* then an empty string is returned.

**string repeat *string count***
Returns *string* repeated *count* number of times.

**string replace *string first last* ?*newstring*?**
Removes a range of consecutive characters from *string*, starting with the character whose index is *first* and ending with the character whose index is *last*. An index of 0 refers to the first character of the string. *First* and *last* may be specified as for the **index** method. If *newstring* is specified, then it is placed in the removed character range. If *first* is less than zero then it is treated as if it were zero, and if *last* is greater than or equal to the length of the string then it is treated as if it were **end**. The initial string is returned untouched, if *first* is greater than *last*, or if *first* is equal to or greater than the length of the initial string, or *last* is less than 0.

**string reverse *string***
Returns a string that is the same length as *string* but with its characters in the reverse order.

**string tolower *string* ?*first*? ?*last*?**
Returns a value equal to *string* except that all upper (or title) case letters have been converted to lower case. If *first* is specified, it refers to the first char index in the string to start modifying. If *last* is specified, it refers to the char index in the string to stop at (inclusive). *first* and *last* may be specified using the forms described in **STRING INDICES**.

**string totitle *string* ?*first*? ?*last*?**
Returns a value equal to *string* except that the first character in *string* is converted to its Unicode title case variant (or upper case if there is no title case variant) and the rest of the string is converted to lower case. If *first* is specified, it refers to the first char index in the string to start modifying. If *last* is specified, it refers to the char index in the string to stop at (inclusive). *first* and *last* may be specified using the forms described in **STRING INDICES**.

**string toupper *string* ?*first*? ?*last*?**
Returns a value equal to *string* except that all lower (or title) case letters have been converted to upper case. If *first* is specified, it refers to the first char index in the string to start modifying. If *last* is specified, it refers to the char index in the string to stop at (inclusive). *first* and *last* may be specified using the forms described in **STRING INDICES**.

**string trim *string* ?*chars*?**

Returns a value equal to *string* except that any leading or trailing characters present in the string given by *chars* are removed. If *chars* is not specified then white space is removed (any character for which **string is space** returns 1, and "\0").

**string trimleft *string* ?*chars*?**
Returns a value equal to *string* except that any leading characters present in the string given by *chars* are removed. If *chars* is not specified then white space is removed (any character for which **string is space** returns 1, and "\0").

**string trimright *string* ?*chars*?**
Returns a value equal to *string* except that any trailing characters present in the string given by *chars* are removed. If *chars* is not specified then white space is removed (any character for which **string is space** returns 1, and "\0").

## OBSOLETE SUBCOMMANDS

These subcommands are currently supported, but are likely to go away in a future release as their functionality is either virtually never used or highly misleading.

**string bytelength *string***
Returns a decimal string giving the number of bytes used to represent *string* in memory when encoded as Tcl's internal modified UTF-8; Tcl may use other encodings for *string* as well, and does not guarantee to only use a single encoding for a particular *string*. Because UTF-8 uses a variable number of bytes to represent Unicode characters, the byte length will not be the same as the character length in general. The cases where a script cares about the byte length are rare.
In almost all cases, you should use the **string length** operation (including determining the length of a Tcl byte array value). Refer to the **Tcl_NumUtfChars** manual entry for more details on the UTF-8 representation.

Formally, the **string bytelength** operation returns the content of the *length* field of the **Tcl_Obj** structure, after calling **Tcl_GetString** to ensure that the *bytes* field is populated. This is highly unlikely to be useful to Tcl scripts, as Tcl's internal encoding is not strict UTF-8, but rather a modified CESU-8 with a denormalized NUL (identical to that used in a number of places by Java's serialization mechanism) to enable basic processing with non-Unicode-aware C functions. As this representation should only ever be used by Tcl's implementation, the number of bytes used to store the representation is of very low value (except to C extension code, which has direct access for the purpose of memory management, etc.)

*Compatibility note:* it is likely that this subcommand will be withdrawn in a future version of Tcl. It is better to use the **encoding convertto** command to convert a string to a known encoding and then apply **string length** to that.

```
string length [encoding convertto utf-8 $theString]
```

**string wordend *string charIndex***
Returns the index of the character just after the last one in the word containing character *charIndex* of *string*. *charIndex* may be specified using the forms in **STRING INDICES**. A

word is considered to be any contiguous range of alphanumeric (Unicode letters or decimal digits) or underscore (Unicode connector punctuation) characters, or any single character other than these.

**string wordstart *string charIndex***
Returns the index of the first character in the word containing character *charIndex* of *string*. *charIndex* may be specified using the forms in **STRING INDICES**. A word is considered to be any contiguous range of alphanumeric (Unicode letters or decimal digits) or underscore (Unicode connector punctuation) characters, or any single character other than these.

## STRING INDICES

When referring to indices into a string (e.g., for **string index** or **string range**) the following formats are supported:

***integer***
For any index value that passes **string is integer -strict**, the char specified at this integral index (e.g., **2** would refer to the “c” in “abcd”).

**end**
The last char of the string (e.g., **end** would refer to the “d” in “abcd”).

**end-*N***
The last char of the string minus the specified integer offset *N* (e.g., “**end-1**” would refer to the “c” in “abcd”).

**end+*N***
The last char of the string plus the specified integer offset *N* (e.g., “**end+-1**” would refer to the “c” in “abcd”).

***M+N***
The char specified at the integral index that is the sum of integer values *M* and *N* (e.g., “**1+1**” would refer to the “c” in “abcd”).

***M-N***
The char specified at the integral index that is the difference of integer values *M* and *N* (e.g., “**2-1**” would refer to the “b” in “abcd”).

In the specifications above, the integer value *M* contains no trailing whitespace and the integer value *N* contains no leading whitespace.

## EXAMPLE

Test if the string in the variable *string* is a proper non-empty prefix of the string **foobar**.

```
set length [string length $string]
if {$length == 0} {
    set isPrefix 0
} else {
    set isPrefix [string equal -length $length $string "foobar"]
}
```